**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий, механики и оптики**

**Кафедра информатики и прикладной математики**

Основы программной инженерии

Лабораторная работа №3

«Модульное тестирование ПО».

 Выполнили:

 Кудряшов А.

Шайхиев А.

Группа 2121

2012 г.

Задание:

С помощью пакета [JUnit](http://www.junit.org/%22%20%5Ct%20%22_blank) провести модульное тестирование программы, реализующей [задание к лабораторной работе #3](http://helios.cs.ifmo.ru/disciplines/iaps%22%20%5Cl%20%22lab3) по дисциплине "Программирование интернет-приложений".  Тестировое покрытие должно быть реализовано для всех классов приложения, тестовые сценарии должны учитывать все возможные варианты попадания или непопадания точек в область на координатной плоскости, а также возможность ввода пользователем некорректных данных.

Требования к тестируемому приложению:

Приложение должно содержать следующие классы:

Класс Mark, представляющий точку с координатами X и Y типа double.

Класс Outline(Target), представляющий область с заданным параметром R, в котором должен быть реализован метод, возвращающий для заданной точки значение 1, если точка входит в область, и 0, если не входит. Попадание на границу области не считается попаданием в область.

Класс Lab2, который получает параметр R типа int в качестве аргумента командной строки. Получение числа из строки реализовать c помощью метода Integer.parseInt().



Описание тестового покрытия:

1. Проверка ввода корректных данных пользователем для задания параметров мишени
2. Проверка адекватной реакции приложение на «выстрелы» вне, на границе и внутри частей области, заданных разными уравнениями
3. Проверка реакции приложения на «выстрелы» в границы между частями заданной области
4. Проверка реакции приложения на «выстрелы», по области в целом,

 формируемые по эталонному уравнению псевдослучайным методом.

Исходный код классов тестов:

Класс№1

**import** **static** org.junit.Assert.\*;

**import** java.util.\*;

**import** org.junit.\*;

**import** org.junit.runner.RunWith;

**import** org.junit.runners.\*;

**import** org.junit.runners.Parameterized.Parameters;

@RunWith(Parameterized.**class**)

**public** **class** TargetInTestingParamClass {

 **boolean** chk;

 Object prmtr,t;

**public** TargetInTestingParamClass(Object prmtr,**boolean** chk){

**this**.chk = chk;

**this**.prmtr = prmtr;

}

@Test

**public** **void** TargetInTest( ){

 **try**{

 t = **new** Target(prmtr);

 **if** ((t.getClass()==Target.**class**)!=chk){

 *fail*("Неверная обработка входных данных");

 }}

 **catch**(Exception e)

 {*fail*("Необработано исключение: "+e.toString());

 };

}

@Parameters

**public** **static** List<Object[]> ParamForTargetGoodInTest(){

**return** Arrays.*asList*(**new** Object[][]{

 { (**int**)2,**true**},

 {"a",**false**},

 {(**int**)-1,**false**},

 {2.5,**false**},

 {(**int**)5,**true**},

 {3455,**true**}});

}

}

Класс №2

**import** **static** org.junit.Assert.\*;

**import** java.util.Random;

**import** org.junit.BeforeClass;

**import** org.junit.Test;

**public** **class** ShotTestClass {

 Mark m1,m2,m3;

 **static** **int** *r*;

 **static** Random *RandomVar*;

 **static** Target *t*;

 @BeforeClass

 **public** **static** **void** MakeDataVar()

 {

 *RandomVar* = **new** Random();

 *r* = (**int**)Math.*abs*(*RandomVar*.nextInt()/2);

 *t* = **new** Target(*r*);

 }

@Test

**public** **void** TestShotFirstArea()

{

 m1 = **new** Mark(-2\**r*,0);

 m2 = **new** Mark(-*r*/2,(-*r*/2)/2+*r*/2);

 m3 = **new** Mark(-*r*/5,*r*/5);

 *assertTrue*(*t*.Shot(m1)==0&&*t*.Shot(m2)==0&&*t*.Shot(m3)==1);

 }

@Test

**public** **void** TestShotSecondArea()

{

 m1 = **new** Mark(2\**r*,2\**r*);

 m2 = **new** Mark(*r*/2,*r*/2);

 m3 = **new** Mark(0.25\**r*,0.25\**r*);

 *assertTrue*(*t*.Shot(m1)==0 && *t*.Shot(m2)==0 && *t*.Shot(m3)==1);

 }

@Test

**public** **void** TestShotThirdArea()

{

 m1 = **new** Mark(2\**r*,-2\**r*);

 m2 = **new** Mark(0,-*r*);

 m3 = **new** Mark(0.25\**r*,-0.25\**r*);

 *assertTrue*(*t*.Shot(m1)==0&&*t*.Shot(m2)==0&&*t*.Shot(m3)==1);

 }

@Test

**public** **void** TestShotBorderAndFreeArea()

{

 m1 = **new** Mark(0,0.25\**r*);

 m2 = **new** Mark(0.25\**r*,0);

 m3 = **new** Mark(-(**int**)Math.*abs*(*RandomVar*.nextInt()/2),-(**int**)Math.*abs*(*RandomVar*.nextInt()/2));

 *assertTrue*(*t*.Shot(m1)==1&&*t*.Shot(m2)==1&&*t*.Shot(m3)==0);

 }

@Test

**public** **void** TestManyShots()

{

 **for**(**int** i = 0;i<1000000;++i)

 {

 Random RandomVar1 = **new** Random();

 Random RandomVar2 = **new** Random();

 **int** x = (**int**)Math.*abs*(RandomVar1.nextInt()/2);

 **int** y = (**int**)Math.*abs*(RandomVar2.nextInt()/2);

 **short** CheckShort;

 **boolean** Check = Math.*pow*(x, 2)+Math.*pow*(y, 2)<Math.*pow*(*r*/2, 2)&&x>0&&y>0||x>0&&x<*r*/2&&y<0&&y>-*r*||y>0&&x<0&&y<x/2+*r*/2||y==0&&x<*r*/2&&x>-*r*;

 **if**(Check)

 {CheckShort = 1;}

 **else**

 {CheckShort = 0;}

 **if**(*t*.Shot(**new** Mark(x,y))!=CheckShort)

 {*fail*("Сбой при беспорядочной стрельбе");}

 }

 }

}

Класс №3

**import** org.junit.runner.RunWith;

**import** org.junit.runners.Suite;

**import** org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.**class**)

@SuiteClasses({TargetInTestingParamClass.**class**,ShotTestClass.**class**})

**public** **class** TestsRunClass {

}

Результаты тестирования:



Вывод:

Во время выполнения лабораторной работы были изучены варианты реализации, основные понятия модульного тестирования и возможности, предоставляемые пакетом Junit 4-ой версии.